ISSUE 1 MAY '82

THE BEEBON

*FX CALLS LISTED
COMPLETE MEMORY MAP
SOUND COMMANDS EXPLAINED
ADVANCED GRAPHICS
BLITZ

£1

THE BEEBON

ISSUE #1 MAY 1982

AVAILABLE BY SUBSCRIPTION ONLY
SEVEN POUND FIFTY PER YEAR

COPYRIGHT NOTICE

The BEEBON is published bi-monthly on the third week of the month. All correspondance should be addressed to:-

THE EDITOR
THE BEEBON
106 THE ALBANY
OLD HALL ST.
LIVERPOOL
MERSEYSIDE
L3 9EP

Any letters which require a reply should be accompanied by a S.A.E. As we are extremely busy those enquiries which are phrased so as to make them simple to answer will solicit the fastest response.

Programs and articles submitted for publication should preferably be typed or computer printout. Clear handwriting is also acceptable. All programs must be on tape or disk and have clear instructions for use and as much supporting documentation as possible. A day time telephone number would be appreciated.

Payment is ten pounds per published page with part pages earning a proportional amount. We are prepared to negotiate for any very exceptional material.

Welcome to our first issue. Unfortunately it has been heavily influenced by the extremely poor, albeit provisional, manual which comes with the machine. It appears to us that it would not have taken much effort to include a 4 or 5 page document with the machine explaining such things as the sound commands. We would be doing a grave diservice to our readership if we did not publish some of the information we have gathered, in the past few months, as the new manual seems to be still in the preliminary proofing stages. We have held over some of the larger programs planned for this month in order to fit in some basic explanations of the FX calls and sound statements.

Talking of information we have found ACORN to be extremely reticent to give details of the machines hardware, being especially paranoid over the video ULA. Protracted efforts to prise information from them has resulted in excuses ranging from "we don't believe you need it"(technical manager) to "I don't remember offhand " and "it isn't written down anywhere" (same conversation with machine operating system author). To be fair we must mention "I haven't a clue. I don't know much about computers"(BBC's software manager).

It does not bode well for the BBC micro if ACORN restrict information to those people they approve of whilst deliberately ignoring those who really need such advanced information. A machine no matter how well designed is nothing without software. We now have all the information needed but only by disassembling the operating system.

We hope ACORN issue the new operating system not only free of charge, but automatically. We also hope they replace all those video ULA's on model A's which do not work on graphics modes 3 to 0.

This editorial may be a little critical but Acorn and the BBC are both large enough and arrogant enough to destroy the future of a superb computer. We must do all we can to protect our investements.

SHAPES

The following program draws some pretty patterns on the screen. Although these are very nice, the important thing about it is the procedure it uses to do it. Taking 8 parameters it draw shapes of any size including squares, triangles and dodechahedrons (if you really must). It can also draw circles and ovals. It allows these shapes to be drawn anywhere on the screen in any colour and size and filled if desired.

The actual PROCEDURE starts at line 100 and may be extracted complete and used in any BASIC program as it is entirely self-contained. Type the program in and RUN it for an idea of what it can do. Model A owners should change the MODE2 statement to MODE5. For something a bit weird change the 7th parameter in the procedure call in line 30 to 3 and let it run for a while.

```
 10 MODE2
 20 REPEAT
 30   PROCSHAPE(RND(20),RND(255),RND(255),RND(80),RND(80),RND(16)-1,3,-1)
 40   UNTIL 0
100 DEF PROCSHAPE(NO%,XORG%,YORG%,XRADIUS%,YRADIUS%,COL%,ACTION%,FILL%)
110 LOCAL M%,DOTS,C,S,X1,Y1,X,Y
120 GCOLACTION%,COL%
130  IF FILL%=TRUE THEN M%=85 ELSE M%=5
140 DOTS=RAD(360/NO%)
150 C=COS(DOTS)
160 S=SIN(DOTS)
170 X1=1:Y1=0
180 FOR I%=1 TO NO%+1
190   X=X1*C-Y1*S
200   Y=X1*S+Y1*C
210   XCOORD%=X*XRADIUS%+XORG%
220   YCOORD%=Y*YRADIUS%+YORG%
230   IF I%=1 THEN MOVE XCOORD%*5,YCOORD%*4:GOTO260
240   IF M%=85 THEN MOVE XORG%*5,YORG%*4
250   PLOTM%,XCOORD%*5,YCOORD%*4
260   X1=X:Y1=Y
270   NEXT
280 ENDPROC
```

The procedure is called as PROCSHAPE(P1,P2,P3,P4,P5,P6,P7,P8) where the parameters are:-

- P1 The number of points in the shape. two will draw a line, 3 a triangle, 4 a rectangle and so on. About 20 are needed for a decent circle.
- P2 The screen X coord of the centre of the shape.
- P3 The screen Y coord of the centre of the shape.
- P4 The radius or width of the shape in the X direction.
- P5 The radius or length of the shape in the Y direction.
- P6 The colour of the shape.
- P7 The mode of action when plotting the shape, exactly as in the first parameter in a GCOL statement.

P8 If set to -1 or TRUE then the shape is filled with colour.
Switch to a graphics mode and play with the procedure by using immediate statements and you soon get the feel of it. It can be speeded up significantly by removing spaces, using multiple statement lines and shortening variable names.

i.e

```
MODE5:PROCSHAPE(30,128,128,100,100,1,0,-1)
MODE5:PROCSHAPE(5,128,128,40,40,2,0,0)
```

For all the masochists out there, the method is an incremental one based on :-

```
X=XORIGIN+XRADIUS*COS(ANGLE)
Y-YORIGIN+YRADIUS*SIN(ANGLE)
```

WHERE THE ANGLE GOES FROM 0 TO 360 DEGREES ANGLEI 0 TO 360 RATIONALISED TO:-

```
NEXTX=XORIGIN+(LASTX-XORIGIN)*COS(ANGLEI)-(LASTY-YORIGIN)*SIN(ANGLEI)
NEXTY-YORIGIN+(LASTX-XORIGIN)*SIN(ANGLEI)+(LASTY-YORIGIN)*COS(ANGLEI)
```

This is repeated for the number of points required with ANGLEINC=360/NO OF POINTS and starting with INITIALX=XORIGIN+RADIUS and INITIALY=YORIGIN. Further modifications allow separate X and Y radius.

MEMORY MAP

The memory map published in the provisional manual is slightly incomplete, to say the least, missing out as it does the addresses of some rather important chips. We present here a complete but still rather basic map which leaves out, for space reasons, the descriptions of the registers within the actual chips in the internal I/O area. If readers write in and ask for it we'll have a full map in the next issue.

ADDRESS	DEVICE	READ	WRITE	DESCRIPTION
0000	RAM	X	X	6502 ZERO PAGE The allocation of space appears to be as follows:- 0-6F BASIC/LANGUAGE ROM 70-8F FREE 90-FF MOS WORKSPACE
0100	RAM	X	X	6502 STACK
0200	RAM	X	X	AS IN PROVISIONAL MANUAL PAGE 224 Except &C00-&CFF is for character definitions 224-255 and not as stated.
4000	RAM	X	X	RAM BANK2 START Present only in model B
8000	ROM	X		SELECTED LANGUAGE ROM Four ROMs may share this part of memory although only one may be enabled at a time.
C000	ROM	X		MACHINE OPERATING SYSTEM
FC00	I/O	X	X	EXTERNAL INPUT/OUTPUT 1 Known as "FRED"
FD00	I/O	X	X	EXTERNAL INPUT/OUTPUT 2 Known as "JIM"

				FRED and JIM are both part of the 1MHZ expansion bus intended for such things as Teletext and Prestel The next 256 bytes of I/O is dedicated to internal use and is known as "SHEILA".
FE00	CRTC	X	X	CATHODE RAY TUBE CONTROLLER This chip has 18 user accessible registers. It handles the position of the display within RAM memory, video format timing, the screen size, the cursor, and the light pen.
FE08	ACIA	X	X	ASYNCHRONOUS COMMUNICATIONS INTERFACE ADAPTER Controls the conversion, in both directions, of parallel and serial data with full formatting and error checking.Used by both the RS423 and cassette interfaces.
FE10	SULA		X	SERIAL UNCOMMITED LOGIC ARRAY Handles the 8 baud rates. Also includes the cassette interface and switches for ACIA to RS423/cassette and the cassette motor relay.
FE18	????			UNKNOWN AT PRESENT
FE20	VULA		X	VIDEO UNCOMITTED LOGIC ARRAY Includes a very fast 64 bit RAM which is used as the colour palette. Also has the system clock generator, video timing, cursor control and display invert logic.
FE30	SEL	X	X	ROM BANK SELECT Selects one of four paged ROMS.
FE40	VIAA	X	X	VERSATILE INTERFACE ADAPTER A This VIA handles the keyboard and sound generator, which has 8 registers of its own. It also provides the timers which run the system clocks and interrupt structure.
FE60	VIAB	X	X	VERSATILE INTERFACE ADAPTER B Provides 16 registers to control the Parallel printer and user interfaces, and also two user programable timers and a shift register.
FE80	FDC	X	X	FLOPPY DISK CONTROLLER Contains most of the logic neccessary to interface and control both 5 1/4 and 8 inch disk drives.
FEA0	ADLC	X	X	ADVANCED DATA LINK CONTROLLER Handles the transmission and receipt of data in a network, in this case the Econet. Has 9 registers through which complex communications protocols are controlled.

FEC0	ADC	X	X	ANALOGUE TO DIGITAL CONTROLLER Has 8 registers through which 4 12 bit analogue to digital converters are controled.Model B only.
FEE0	TUBE	X	X	EXTERNAL INPUT/OUTPUT CONTROL The control for the 2MHZ expansion bus called the TUBE are situated here. Provides the means of communication with the second (optional) proccesor.
FF00	ROM	X		MOS VECTORS All of the MOS interrupt and user vectors are situated here.

COLOURFUL CHARACTERS

Although it is extremely easy to redefine the character set with VDU23, control over the colour of individual pixels within a character is not directly catered for by the VDU drivers. The following short program illustrates a simple solution.

```
10 MODE 5:VDU 5
20 MOVE 512,512:PRINT "A"
30 GCOL 0,1
40 MOVE 512,512:PRINT "I"
```

You need only define seperate characters for each of the colours used then overprint them to get the desired result.

MOS TITBITS

Two MOS commands:-

*MOTOR1	Turns the cassette motor on
*MOTOR0	Turns the cassette motor off
*TV255,1	Moves the entire display down by one line on the T.V. screen and switchs on interlacing.
*TV255,0	Just moves the display.

Other values for *TV move the display by different amounts. The point of it is that many T.V.'s cannot display the top line at all which is bit inconvenient if yours is one of them.

Interlacing has the effect of drawing everything on the screen twice on alternate scan lines. This makes text a lot more dense but often causes the screen to flicker slightly.

COMPETITION

There are two commands which we don't have information on as yet. They are *SPOOL and *DEBUG . The first correct (and complete) explanations of each which we receive, before the new manual appears with explanations of them, will win the senders something nice, a credit voucher probably.

SET IT UP

It is often useful to set certain options on your machine everytime you power-up to suit your environment and programming methods. Some examples are using *TV because you lose the top line on your T.V., setting up the serial interface to drive your printer, and using the filter option to remove line feeds, and perhaps most importantly setting the user-definable function keys to some useful functions.

The following program which we shall call SETUP enables all these things, or anything else you fancy, automatically by means of the cassette operating system command *EXEC. This simply substitutes the input from the cassette for the normal keyboard input.

Simply type the program in as it appears and RUN it whilst recording onto a blank cassette. When the program is finished you will have a data tape which you can use to set up your machine. To do so type *EXEC "SETUP" and play it when you first switch on.

```
L.
   10 REM Open file
   20 X=OPENOUT("SETUP")
   30 REPEAT READ A$,A
   40   REM Reverse string A$ and put it into string B$
   50   B$=""
   60   FORXX%=LEN(A$) TO 1 STEP -1
   70     B$=B$+MID$(A$,XX%,1)
   80     NEXT
   90   REM Output sentence to tape
  100   PRINT#X,B$
  110   REM Output a carriage return
  120   BPUT#X,A
  130   REM End is always the last text
  140   UNTIL A$="END"
  150 CLOSE#X
  160 END
  170 REM PLACE SET UP STATEMENTS AND COMMANDS IN DATA STATEMENTS HERE
  180 DATA 170DATA P."THIS IS THE FIRST LINE",&0D
  190 DATA P."THIS IS THE SECOND LINE",&0D
  200 DATA*KEYOIT WORKED,&0D
  210 REM ALWAYS END WITH THIS STATEMENT
  220 DATA END,&0D
```

You may customize the set up options by typing in your own DATA statements each of which follows the format :-

```
<line number> DATA <BASIC, or MOS command or statement>,&0d
  .....
  ..... repeat the above as many times as necessary
  .....
<line number>DATA END,&0D  .
```

The angle brackets (<>) mean that their contents are required but the exact format of these contents is up to the user. The END statement tells the program that it is the end of the data, and the &0D's are carriage returns which are necessary to ensure the command or statement is obeyed.

SPECIAL EFFECTS

The Machine Operating System (M.O.S.) provides the user with a set of system calls for a variety of functions concerning housekeeping and special effects. They are naturally called with *FX <parameters>, effects-get it?. They are a bit like COS commands in that they need the "*" character to signal to BASIC that they are meant for the M.O.S. to deal with, amd they cannot have anything following them on the same line.

They may take up to 2 parameters in addition to the FX function number. We present below a fairly detailed description of those that we are aware of at the present time. Please note that a large number of them, marked by a # character in the first column are only available on the new vers 1.0 M.O.S., and as we do not yet have a copy of it, are untested and unguaranteed.

The information given is in many cases provisional so do make sure you try them out before using them for anything important. We cannot vouch for the correctness of the function numbers not implemented on vers 0.1 as none of the information we have originates from ACORN directly and is in no way "official". We can however be fairly sure that all calls given as unique to MOS vers 1.0 will be represented when it becomes available with any incorrectness relating only to the exact parameters required.

If you are not sure which version you have try typing *FX0.

To use them from Basic type *FX <FX number>,<parameter 1>,<parameter 2> in either command or deffered modes. An alternative which is useful for those calls meant specifically for machine code usage is to set A% to the function number, X% and Y% to any parameters and use RESULT%=USR(&FFF4). See the manual for more details of USR if this is unclear.

Most of these calls are available from machine code programs by loading the FX function number into the accumulator, the first parameter into the X register, the second into the Y register and calling OSBYTE at &FFF4. In many cases parameters are not needed or are optional. You must set the decimal mode flag before the call (??) and upon exit the contents of the registers will be indeterminate except where a result is returned.

*FX0	Returns the operating system version number.
#*FX2,X	Select input device?. The exact effect not known.

##*FX3,X Selects the devices used for the output stream, where X is a four bit (0-15) number as follows:-

bit 0 If 0 then output sent to VDU drivers
bit 1 If 1 then output sent to RS 423
bit 2 If 0 then output sent to selcted printer
bit 3 If 0 then output sent to SPOOL file

This allows the same output to be sent to several devices simultaneously. SPOOLing is a very useful thing which should allow, for instance a file to be SPOOLed to the cassete recorder whilst it is being printed. This option is set up with the command *SPOOL which we have'nt yet got working properly.

*FX4,X When X is 1 the copy and cursor keys generate ASCII codes 135 to 139, a program can then use GET or INKEY to detect if they have been pressed. When X=0 the keys are reset to their normal editing functions.

*FX5,X Select printer driver as in provisional manual page 194.

*FX6,X Cause's the printer driver to filter out all occurences of ASCII character X. This can be extremely useful if your printer automatically supplies a LF character whenever it receives a CR thus causing double spaced output.*FX6,10 will cure this problem (10 is the ASCII code for LF). Another use is if you are using the TRACE command in BASIC, to aid debuging, with the printer switched on because the program uses graphics and the line numbers are not being shown. Every time the program switches MODE or does a CLS the printer will do a form feed (extremely annoying). *FX6,12 will cure this.

*FX7,X Sets the RS423 receive baud rate as in the provisional manual on page 194.

*FX8,X As above, for send baud rate.

*FX9,X Makes the duration of the first flashing colour's period of visibility equal to X in hundredths of a second.

*FX10,X As *FX9 but for the second colour's period.

*FX11,X Sets the delay before the keys repeat to X in hundreths of a second. If X is 0 then the auto repeat is turned off.

*FX12,X Sets the keyboard auto-repeat speed to X in hundredths of a second. If X is 0 then the auto repeat is reset to back to normal.

##*FX13,X AND *FX14,X are very complex and are concerned with the control of system interrupts from machine code. We will have a seperate article on these interupts and calls at a later date.

*FX15,X Empties the various buffers selected by X as follows:-

0 all buffers
1 keyboard type ahead buffer
2 RS 423 input
3 RS 423 output
4 printer output
5 sound channel 0
6 sound channel 1
7 sound channel 2
8 sound channel 3

Although all are useful *FX15,1 is absolutely essential

when using the INKEY statement in programs. The options effecting the sound channels will cause all sounds currently playing to be cut off instantly.

*FX16,X Selects the number of ADC channels to be enabled where X is the number (0-4) required. 0 will disable all channels. The less channels there are actually enabled the quicker the converted results will be available from the required channels to your program.

#*FX17,X Forces the start of a conversion on ADC channel X

#*FX18,X(?) Resets the user defined function keys to their normal use. Used after *FX227/228.

#*FX19 Causes a wait for the start of the television field synchronization pulse. One problem with fast character graphics used from BASIC is the amount of flickering which takes place, which is caused by the graphics being drawn whilst the CRT's electron beam is off screen during horizontal or vertical retrace. By synchronization with this pulse it is possible for the flickering to be eliminated.

#*FX20,X Causes character X if redefined (with VDU23) to be reset back to normal, thus freeing the RAM used for its redefinition and enabling it's display as the normal ASCII, and default, representation

There is a large gap here of numbers which have no functions that we are aware of at present.

*FX124 Resets the effect of pressing the ESCAPE key. (we haven't a clue as to the usefulness of this function)

*FX125 Has the same effect as pressing the ESCAPE key????

*FX126 Acknowledge the detection of an ESCAPE condition.

*FX127,X Used from a machine code program, with a previously opened file channel number in the X register, it checks for the end of file condition. On return the X register does not equal zero if the end of the file has been reached.

*FX128,X Performs the same function from a machine code program as the BASIC function ADVAL. On entry register X holds the required channel number (1-4). The result is a 16 bit value in X and Y with X hilding the lsb. *FX128,0 may be used to check the fire buttons on joysticks or paddles as register X is returned with their status in it's two least significant bits.

*FX129,X,Y This call provides the same function from machine code as INKEY. On entry X and Y hold the time delay in 100ths of a second with the lsb in X. On exit the carry flag equals zero if the accumulator contains a valid character. If the carry flag is set then A contains either &80 indicating a time-out condition or &80 if ESCAPE has been pressed in which case *FX126 must be used to acknowledge it.

*FX130 Returns the high order 16 bits of the machines (emulated) 32 bit address. This is used for filing systems and expansion.

*FX131 Returns the first location not used by the M.O.S. in X and Y with the lsb in X.

*FX132 As above but returns the lowest address used by the current

display MODE.

*FX133,X As *FX132 but for the MODE whose number is in X upon entry.

*FX134 Returns in X and Y the position of the text cursor on the screen as coordinates.

##*FX135 Reads the character at the current cursor positon and returns it in X. X will equal 0 if the character cannot be recognised.

*FX136 Reads the last known X and Y coordinates of the light pen into the X and Y registers.

*FX137,X Controls the cassette motor. If X is 1 then it is switched on else if X is 0 then it is switched off. Directly equivelant to *MOTOR

##*FX138 Used to insert charcters into the keyboard buffer thus allowing, for instance, a BASIC program to edit itself.

*FX139 Allows several options when using the cassette filing system. *FX139,1,0 turns off screen messages, *FX139,2,2 aborts the current file operation upon detection of an error. Other options are available but are not known at present.

*FX140 This is exactly equivalent to *TAPE as on page 195 of the provisional manual.

##*FX141 Similar to *FX138 this call allows a program to directly insert characters into the machines buffers. It is followed by two numbers the first of which is the buffer number and the second is the ASCII character to be placed.The buffer numbers are as follows:-

1. KEYBOARD
2. RS423 RECEIVE
3. RS423 TRANSMIT
4. CURRENT PRINTER OUTPUT
5. SOUND CHANNEL 0
6. SOUND CHANNEL 1
7. SOUND CHANNEL 2
8. SOUND CHANNEL 3

From machine code X takes the buffer number, and Y the character.

##*FX142 and *FX143 probably exist but details are unknown at present.

*FX144 Is directly equivalent to *TV mentioned elsewhere in this magazine.

##*FX145 Removes a charcter from a buffer. The opposite of *FX141

We are at present unsure of the next six calls *FX146 to *FX151. It is possible that they have been condensed and moved to *FX252 to *FX254. They will be assumed to have moved. We have no information on any more calls until *FX226 and then the exact numbers which will be used are unreliable. As all of the following calls are only implemented on the new (and at present unreleased) version of the MOS, this unreliability should not matter and those given should be taken as a general guide to the facilities which will become available at some point in the future.

##*FX226 Cancells the vdu queue.

##*FX230,X Causes the function keys to take on ASCII codes 0 to 19 plus an offset X. The codes 10+X to 19+X are generated by holding down the shift key whilst the function key is pressed. The offset X would normally be greater than 128 thus make the Teletext control codes available directly

from the keyboard.

##FX240,X With X=1 the ESCAPE key would return the ASCII code &1B when pressed and with X=0 the ESCAPE key would assume its normal function.

The next 4 calls are related to the generation of sound from machine code. It would appear that whilst the various parameters are set up by *FX calls the actual sound is initiated by VDU7. In other words these calls modify the standard bell sound.

##FX241,X Selects the sound channel as X (0-3).

##FX242,X The interpretation of this call depends on bit 7 of X. If it is 0 then X is the envelope number. If it is 1 then the following is used:-
bits0 and 1 are for synchronization control
bit 2 is for flush control
bits 3 to 6 control the amplitude of the sound

##FX243 Selects the frequency of the "bell".

##FX244 Selects the duration of the "bell".

##FX245 Returns in X the setting of some links on the keyboard which determine the default display mode and wether the machine "boots" from disk on power-up. Details of these links are given elsewhere.

##FX252 Used to access the I/O area named FRED

##FX253 Used to access the I/O area called JIM

##FX254 Used to access internal I/O called SHEILA

All three of the above have the offset in X and the number to be olaced in the Y REG. On exit the X reg will contain the value at that location. Further information on these I/O areas is available in the memory map printed in this magazine.

##FX255 This call returns a value in the X reg which is interpretated as follows. If bit 7 =1 then the computer is a model B, if bit 0=1 then the last reset was a "hard reset"else it was a soft reset.

We are aware that most of this information will be available within the new manual, but at the time of writing this manual looks a long way from being finished and alternative explanations are always useful in the computer world.

CRTC's and RAM

The CRTC (cathode ray tube controller) is a very useful and programmable chip. To demonstrate it's ability to find it's display RAM from anywhere you care to tell it to, type in this short statement exactly as shown:-

```
MODE5:?&FE00=12:?&FE01=0:?&FE00=13:?&FE01=0
```

You are now looking at a graphical representation of the lower 10k of memory. The little changing bits near the top are the system clocks. A lot of other locations are also being altered but they are changing state so fast you can't see them. Notice the chequerboard pattern of the uninitialised memory and also the basic program area in

the middle.

What you actually did was to persuade the CRTC that it's display memory starts at location zero. To get out of it press BREAK. Don't forget OLD if you have a program typed in that you want to keep.

MORE GRAPHICS

The following program was written to specifically demonstrate some of the BBC micro's more unique features. Type it in as shown, but model A owners should omit all REMarks and comments. Remember to save it to cassette before running it, as if any of the assembler portion is mistyped it could destroy the program when run.

```
 10 REM VERY USEFUL WHEN WRITING GRAPHICAL PROGRAMS
 20 ONERRORGOTO880
 30 MODE7
 40 REM EXAMPLE OF DOUBLE HEIGHT TELETEXT CHARACTERS
 50  PRINT''''''''''CHR$141"                    PATTERNS"'CHR$141;"                    PATTERN
S"
 60 REM WAIT A WHILE
 70 FOR XX%=0 TO 10000:NEXT
 80 REM ASSEMBLY NECESSARY FOR PROGRAM TO RUN
 90 PROCASSEMBLE
100 REM WAIT 10 SECONDS FOR KEYPRESS
110 PRINT"TOUCH ANY KEY":A=INKEY(1000)
120 REM START NEW RANDOM NUMBERS WITH TIME TAKEN SO FAR
130 A=RND(-TIME)
140 REM CHANGE TO MODE 5 FOR MODEL A
150 MODE2
160 REM MAIN PROGRAM STARTS
170 GCOL0,129:CLG
180 FOR X%=0 TO 40
190   REM DRAW AND FILL A RECTANGLE
200   PROCBOX
210    REM DRAW SOME RANDOM LINES IN IT WITH FULL CLIPPING
220   PROCLINES
230   NEXTX%
240 REM MACHINE CODE ROUTINE TO CYCLE THROUGH COLOURS
250 PROCFLASH
260  REM END OF MAIN PROGRAM
270 GOTO150
280 DEF PROCBOX
290 LOCAL leftx%,bottomy%,rightx%,topy%
300 REM DEFINE A GRAPHIC WINDOW AT RANDOM
310 leftx%=RND(960):bottomy%=RND(768)
320 rightx%=leftx%+RND(1280-leftx%):topy%=bottomy%+RND(1024-bottomy%)
330 VDU24,leftx%;bottomy%;rightx%;topy%;
340 REM NOW FILL IT WITH A RANDOM COLOUR
350 GCOL0,RND(16)+127:CLG
360 ENDPROC
370 DEF PROCLINES
380 REM DRAW RANDOM LINES OF RANDOM COLOURS
390 REPEAT GCOL RND(5)-1,RND(16)-1
```

```
      BEEBON MAY 1982


400    PLOT5,RND(1279),RND(1023)
410    REM A RANDOM NO OF TIMES
420    UNTIL RND(50)=1
430 ENDPROC
440 DEF PROCFLASH
450 LOCAL CC%
460 REM CYCLE THROUGH COLOUR CHANGES 5 TIMES
470 FOR CC%=0 TO 5
480     REM EXECUTE MACHINE CODE ROUTINE AT HEXIDECIMAL A00
490     CALL&0A00
500    NEXTCC%
510 ENDPROC
520 DEF PROCASSEMBLE
530 REM MAKE VARIABLES LOCAL-GOOD PROGRAMMING PRACTISE
540 LOCAL oswrch%,option%,counter%
550 REM DEFINE VARIABLES
560 oswrch%=&FFEE:counter%=&70
570 REM LOOP THROUGH ASSEMBLER SOURCE TEXT TWICE WITH option% EQUAL TO ?AN
580 FOR option%=0 TO 3 STEP 3
590    REM PLACE CODE IN RS423 RECIEVE BUFFER-FAIRLY SAFE.CAN BE CHANGED TO !
O
600    P%=&A00
610    REM START ASSEMBLER WITH [ FOR OPT SEE PAGE 131 PROVISIONAL MANUAL
620    [OPToption%
630    LDA #10 \NUMBER OF TIMES THE ROUTINE EXECUTES
640    STA counter% \SAVE NUMBER IN ZERO PAGE SCRATCHPAD
650    .OUTER LDY #0 \WILL BE 255 TIMES THROUGH COLOUR CHANGES
660    .LOOP LDA #19 \VDU CODE FOR PHYSICAL COLOUR CHANGE
670       JSR oswrch% \SEND CODE TO VDU DRIVERS
680       LDA counter% \SEND LOGICAL COLOUR
690       JSR oswrch%
700       TYA \SEND NEW PHYSICAL COLOUR
710       JSR oswrch%
720       LDA #0 \FOR THREE REQUIRED DUMMY ZEROS
730         LDX #3 \COUNTER
740         .DUMMYS JSR oswrch% \SEND A ZERO
750         DEX \COUNT DOWN BY ONE
760         BNE DUMMYS \AND REPEAT IF NOT ZERO
770           DEY \COUNT DOWN COLOUR CHANGES.WILL GO FROM 0 TO 255 THE FIRST T
E
780           BNE LOOP \REPEAT MAIN LOOP IF NOT ZERO
790             LDA #20 \RESET DEFAULT COLOURS
800             JSR oswrch% \SEND CODE
810             DEC counter% \LOGICAL COLOUR COUNTDOWN. THIS VARIABLE IN MEMO
820             BNE OUTER \REPEAT IF NOT ZERO
830               RTS \ COME HERE WHEN FINISHED SO RETURN TO BASIC
840                \ SWITCH OFF ASSEMBLER WITH ] THEN GO THROUGH TEXT AGAIN IF
ECESSARY
850                ]:NEXT
860          ENDPROC
870          REM IN CASE OF ERRORS WHILST IN GRAPHICS MODE AND FIDDLING WIT
COLOURS
880          MODE7:PRINTERL:REPORT
>
>
>
```

CURSORS

Is that nasty cursor blinking all over your nicely drawn graphics? To switch it off use:-

```
?&FE00=10:?&FE01=32
```

If on the other hand you like having the cursor popping up all over the place but just don't like the way it looks use:-

```
?&FE00=10:?&FE01=0   Gives a solid block
?&FE00=10:?&FE01=64  Gives a quickly blinking block
?&FE00=10:?&FE01=96  Gives a slowly blinking block
```

You may have to fiddle around a bit with it to prevent the MOS from reseting the cursor all of the time.

LOCAL VARIABLES

The concept of local variables can be a little confusing to those not used to them. This short program demonstrates their major properties.

```
>LIST07
>L.
   10 A$="FRED"
   20 PROCOUT("JIM")
   30 PROCOUT(A$)
   40 B$="TONY"
   50 PROCOUT(B$)
   60 PRINTB$
   70 END
   80
   90 DEF PROCOUT(A$)
  100 LOCAL B$
  110 B$="JOHN"
  120 PRINTA$,B$
  130 ENDPROC
>
>RUN
JIM         JOHN
FRED        JOHN
TONY        JOHN
>
>
```

If you don't understand it then take consolation in the fact that I don't either and I wrote it!

LETTERS

Because this is the first issue there are no letters. We would

like, however, to encourage our readers to write in about anything and everything. So we'll give T-shirts to all those whose letters are published.

LINKS

If you look at the circuit board on which the keyboard is mounted you will see, directly below the DELETE key a set of 16 holes for eight soldered links. The function of these links is as follows:-

5-8 RESERVED FOR FUTURE USE
4 FORCES AN AUTO-START FROM DISK
1-3 SELECTS THE DISPLAY MODE FOLLOWING RESET

Because there are no links fitted as standard the default with links 1-3 left open must be MODE 7 and no auto start. We haven't tested this information as yet so if you fit some links and nothing happens then it will be because MOS vers 0.1 doesn't support this feature.

PRINTER STROBE

Acorn have, on the machines we've seen, tied the parall interface strobe line to earth and buffered it. This means that the strobe line goes active high the opposite of that expected by most of the world's printers including the popular Epson brand. If your Epson refuses to work then the following modification has worked for us, but be warned we do not accept responsibility for any damage you may do to your machine whilst affecting modifications we suggest. It's your decision and your responsibility. Whatever you do don't go poking about inside the case with the power on. It could be fatal to you and it certainly won't do your poor computer any good.

Find IC 27 which is on its own in the middle of the left side of the board. It should be marked with 7438 somewhere. If you can't find this chip then forget the whole idea. There should be a transistor directly below IC 27. If this is not there but a metal link block is, swap the link over as this will be all that is neccessary to invert the strobe. The modification will then be completed. If it still doesn't work then it isn't the strobe.

If the transistor is there then you remove IC 27 and lift pins 4, 5, and 6. These are pins 4, 5, and 6 down from the left hand side. Make sure that when you re-insert the chip these pins will not make contact with the socket. Now take a small piece of bare wire and link holes 5 and 6 together in the socket. These are the same holes that pins 5 and 6 of IC 27 came from. Re-insert IC 27 the right way round making sure the lifted pins make no electrical contact with anything. If your printer still doesn't work check that you did everything correctly then reverse the process and have a good cry

NASTY NOISE

Whilst on the subject of hardware modifications and before we get to sound.

The other day I was in the LIVERPOOL COMPUTER CENTRE who are also called MARCH COMMUNICATIONS at times. Whilst I was talking to sales director Tim Best about their involvement with the BBC micro on the sales and hardware front a familiar brown box was delivered. Ah! cried hardware genius Steve Lavanche "my little Beebon has arrived". Steve is

an expert with the Acorn Atom.

We quickly got the machine up and running and as soon as it was switched on Steve noticed the noise enamating from the loudspeaker.This noise is a design fault which Acorn haven't yet cured. Out came the oscilliscope, and notwithstanding that this was his first contact with the machine and his lack of a circuit diagram, 5 minutes later Steve had fixed it completely. He is now well on the way to becoming a BBC micro expert, which is nice for us because he promises to pass on, through this mag, any little mods he dreams up.

The mod is very simple and well worth doing. Make sure all power is off. Locate the sound generator chip in the bottom left hand corner (an SN74689). This is the second one in from the left and will be socketed. To the left is a group of resistors. Look at the markings on the circuit board and find the one labelled R11. Take a small tantalum bead capacitor, 1 micro farad is ideal, and atach one end of it to the uppermost lead of R11. Attach the other end to the minus 5 volt lead, from the power supply, where it joins the main circuit board. MAKE ABSOLUTELY CERTAIN THAT IT IS THE NEGATIVE END OF THE CAPACITOR WHICH JOINS THE POWER SUPPLY. Switch on again and the buzzing noise should of disapeared completely. If you have not understood any part of this paragraph then don't bother trying the mod. Get someone else to do it for you. And don't blame us if you damage your machine, all hardware mods we give are carried out at your own risk.

SOUND ADVICE

Every issue we shall have something on the sound commands. For this month we shall publish a reference guide to their exact format as they were missed out of the provisional manual completely. In future we shall be runnning a small competition to find the best sound effects. The top 3 or 4 will be published and earn their authors a much coveted BUG-BYTE t-shirt whilst the best one will also get a ten pound credit voucher. All submissions should be on cassette if they are of any great length and should be interesting and/or amusing.

The sound commands are all based around the interupt structure of the MOS. Whatever the computer is doing it is interupted at precise time intervals. It then reads the keyboard, the ADC, increments the clocks and so on. At the same time it will update the sound generator according to any sound commands which you may have given. This means that all audio output is transparent, and will not cause the program to halt whilst the processor deals with them.

There are four channels 0 to 3, with channel 0 producing noise.When a sound command is given it is placed in the queue for the specified channel and sounded when any previous sounds for that channel have been dealt with. Each sound may be shaped by an envelope which defines both a frequency and volume envelope for it. This allows the most complex affects and the immitation of almost any musical instrument. The two relevant comands are called, logically enough, SOUND and ENVELOPE. They are extremely complex, and as we don't have much room in this issue, an attempt at a tutorial on their use will be left for a later issue. We shall however give as precise a definition of their use as possible so that you can at least experiment.

The major BASIC statement concerned with sound is quite naturally SOUND P1,P2,P3,P4 with the parameters as follows:-

P1. Is a 4 digit hex field of four parameters
digit 1 gives a channel number 0-3.Channel 0 is noise.
digit 2 gives real or dummy 0 or 1.If 1 the note is a dummy and although it is queued in the normal manner it does not sound.This allows the previous sound's release segment, which continues after the duration is finished, to continue.
digit 3 gives event synchronization 0-3. If the value is 0 the note will execute as normal but if, for instance, the value is 2 the sound will wait for two more notes with digit 3 set to 2, and then all 3 sounds will execute simultaneously.This allows chords of up to four voices.
digit 4 gives the effect on the queue. When 0 as is normal the sound is queued in the channel given by digit 1 and sounded as soon as the channel is free. When 1 the sound flushes the buffer ahead of it and is played immediately no matter what the state of the queue is.
Digit 1 is the leftmost digit and the only one not optional.

P2. If it is negative then it defines the volume number (0 to -15) and if positive it gives the number of the envelope to be applied to the sound (1-4 on vers 0.1).

P3. Is the pitch 0-255. If the note is using a pitch envelope then the increments for that envelope are added directly to P3 which should be taken as a base frequency. If the channel is 0 the noise channel then P3 is interpreted somewhat differently:-
bits 0-1 =0 gives a period of 4.
=1 gives a period of 2.
=2 gives a period of 1.
=3 period is set by channel one.
bit 2 =1 generates pseudo-random noise.
=0 produces a pulsewave.

P4. Is the duration 0-255 in 50 milli second units. This is also the total duration of any envelope used in P1. It applies to both the volume and pitch envelopes in that the all timing increments are derived from it.

Some simple examples are:-

```
SOUND 1,-15,100,20
SOUND 0,-6,2,100
```

The ENVELOPE command is even more complex and has a pitch envelope which is in three seperate parts for which you may define the amount of change per time time period and it's length. Their is also a full ADSR volume envelope. This command takes the form:-
ENVELOPE P1,P2,P3,P4,P5,P6,P7,P8,P9,P10,P11,P12,P13,P14

and the parameters are:-

P1 ENVELOPE NUMBER 1-4 this is the number P2 in the sound command refers to.

P2 SCALING FACTOR 0-127 gives the scaling factor for the rest of the envelope in one hundredths of a second and is the speed at which all of the envelope increments are applied. If bit 7 of this paremeter is 0 then the envelope will be reset and repeated when it finishes and if 0 it will end.

P3 INCREMENT FOR PART1 -128 - +127 How much the note rises or falls per time interval. A value of -1 would reduce the note by 1/4 semi-tone per time interval.

P4 INCREMENT FOR PART2 -128 - +127 as P3

P5 INCREMENT FOR PART3 -128 - +127 as P3

P6 NUMBER OF INCREMENTS FOR PART1 0-255 The total number of times the note is updated for part1.

P7 NUMBER OF INCREMENTS FOR PART1 0-255 As P6

P8 NUMBER OF INCREMENTS FOR PART1 0-255 As P6

P9 ATTACK RATE 1-127 The amount the volume rises to reach the level set by P13 per time interval.

P10 DECAY RATE -127 - +127 The amount the volume decays by each time interval to the level set by P14.

P11 SUSTAIN RATE -128 - 0 The amount the volume falls by until the duration of sound set by P4 of the SOUND command is reached.

P12 DECAY RATE -128 - 0 The amount the volume drops by to reach zero at the end of the sound.

P13 VOLUME LEVEL 1 The absolute volume level for the attack portion of the ADSR envelope.

P14 VOLUME LEVEL2 The second absolute volume level for the ADSR envelope.

We have been told that the new (500 page) manual will contain more than 30 pages devoted to these commands and we wouldn't be suprised if everybody still found them difficult to use after reading it We now give some examples and we'll come back to the correct use of these commands in a later issue. If anybody would like the job of writing a tutorial on them then please, please get in touch.

```
>
>L.
    5 X%=100
   10 ENVELOPE1,3,0,0,0,0,0,0,120,-60,-5,-3,120,70
   20 ENVELOPE2,3,1,0,0,100,0,0,120,-60,-5,-3,12,70
   25 REPEAT G=INKEY(X%):X%=X%-5
   30   SOUND1,2,X%,3
   40   SOUND0,1,X%,3
   50   UNTIL0
>
>
>
    5 X%=10
   10 ENVELOPE1,1,2,-2,5,30,25,15,50,0,0,0,120,0
   15 REPEAT X%=X%+10
   20   SOUND1,1,X%,5
   30   UNTIL0
>
```


CONT P22

BLITZ puts you in the place of a wartime bomber pilot flying over a city. Your plane is somewhat damaged and gradually losing height. To make a safe landing you have to level the city first, to create a runway.

The program has become something of a standard game of late. Acknowledgement should be given to Steven Skipp of Birmingham whose version for the Atom was the first which came to our attention - the following version is completely original, however.

To play the game, you use the SPACE bar to drop bombs. You will find that (normally) there will be anti-aircraft guns firing at you, so it is a good idea to knock out these with your bombs fairly early on. If not they might get you first ! You can try to dodge the shells by pressing the M key to dive down. However this loses height and so should only be used in an emergency. As it loses more and more of its load of bombs, the plane gradually speeds up. It IS possible to level the city before crashing into it but you should find this feat fairly challenging. If you succeed you will be rewarded with a repeat performance on a larger city. Our top score's just over 2000 but you should be able to beat that. Good luck !

```
>L.
  10 REM**********************************
  20 REM       Blitz
  30 REM By Barbara and Tony
  40 REM**********************************
  42 GOSUB9800
  45 MODE 5
  50 REM Define Characters
  60 VDU 23,224,192,224,224,240,255,255,15,31:VDU 23,225,0,0,0,60,255,255,192,0
  70  VDU 23,226,255,153,153,255,255,153,153,255
  80 VDU 23,227,124,16,124,124,124,124,56,16
  85 VDU 23,231,153,90,60,255,24,60,90,153
  90 VDU 23,228,192,224,112,56,28,14,63,255
  92 VDU23,230,0,0,16,56,28,8,0,0
  94 VDU 23,229,255,255,255,255,255,255,255,255
  95 VDU5:SCORE=0:@%=5
  96  HI=7
  98 DIM A(19)
 100 X=0:Y=900:SPEED=25
 110 BM=0:BC=0
 120 GOSUB3010
 130 GOSUB 8510
 200 GCOL3,1:GOSUB 1110
 210 IFBM=0:A$=INKEY$(10):IF A$=" " THEN GOSUB 4110
 214 M=0
 230 IF BM>0 THEN GCOL3,3:GOSUB 5010:GOSUB 5510
 235 IF BM>0 THEN GCOL3,3:GOSUB5010
 240 IF BC>0 THEN GCOL3,3:GOSUB 7510:GOSUB 8010
 245 IF BC>0 THEN GCOL3,3:GOSUB 7510
 250 IF BC=0 THEN GOSUB 7010
 300 GCOL3,1:GOSUB 1110
 310 A$=INKEY$(5):IF A$="M" THEN Y=Y-4
 390 GOSUB 2010
 400 GOTO 200
1000 REM Subroutine to print plane at position x,y
1110 MOVE X,Y:PRINTCHR$224;CHR$225:RETURN
2000 REM Subroutine to move plane
2010 X=X+SPEED:IF X>1289THEN X=0:Y=Y-32
2012 IF X>1120 AND Y<70 THEN GCOL0,1:GOSUB 1110:FOR Z=0 TO 3000:NEXT Z:HI=HI+1:
CLS:GOTO 100
```

```
2014 TX=(X +120)MOD1280
2015 P=POINT(TX,Y-20):IF P >0 THEN GOTO 6510
2020 RETURN
3000 REM Subroutine to draw Bug City
3010 PRINT TAB(0,31);:GCOL0,3:FOR Z=0 TO 19:PRINT CHR$229;:NEXT Z
3015 FOR Z=0 TO 19
3020   R=RND(HI)-1
3030   IF R=0 GOTO 3100
3040   FOR W=1 TO R
3045     GCOL0,2
3050     PRINT TAB(Z,31-W);CHR$226
3060     NEXT W
3062   A(Z)=0
3064   GCOL0,3
3065   IF RND(40)<HI THEN PRINT TAB(Z,31-W);CHR$228:A(Z)=(W+1)*32
3100   NEXT Z
3200 RETURN
4000 REM Routine to drop Bug Bomb
4110 IF BM>0 THEN RETURN
 15 SPEED=SPEED+1
4120 BM=1:BX=X-X MOD64+68:BY=Y-Y MOD32 +2:IF BX>1279 THEN BX=0
4122 GCOL3,3:GOSUB 5010
4125  RETURN
5000 REM Routine to draw Bug Bomb
5010 MOVE BX,BY:PRINT CHR$ 227:RETURN
5500 REM Routine to move Bug Bomb
5510 BY=BY-32
5512  IF POINT(BX+24,BY-28)>0 THEN GOTO 6010
5520 RETURN
6000  REM Routine to explode Bug Bomb
6010 BM=0:GCOL0,0
6015 IF BY<36 THEN RETURN
6020 MOVE BX,BY:IF POINT(BX+32,BY-24)=3 THEN A(BX/64)=0
6025 PRINT CHR$ 229
6030 BY=BY-32
6034 GOSUB 8500
6035 GCOL0,3:SCORE=SCORE+10:GOSUB 8500
6036 SOUND0000,3,50,4
 40 IF RND(3)>1 THEN GOTO 6015
6100 RETURN
6500 REM Death of the Bug plane
6510 GCOL0,0:GOSUB 1110:X=(X+64)MOD 1280:GOSUB 9020
6511 FOR W=0 TO 20:FOR Z=3 TO 0 STEP-1:GCOL0,Z:FOR Q=0TO50:NEXT
6512      MOVEX,Y:PRINTCHR$231
6520       NEXT Z:NEXT W
6999 INPUT Z$:RUN
7000 REM Subroutine to fire Bug cannon
7010 IF BC>0 THEN RETURN
7020 Q=RND(20)-1
7030 IF A(Q)=0 THEN RETURN
7040 CX=Q*64:CY=A(Q)
7045  BC=1:GCOL4,3:GOSUB 7510
7050  RETURN
7500 REM Routine to draw in Anti Bugcraft fire
7510 MOVE CX,CY:PRINTCHR$230
7540 RETURN
8000 REM Routine to move Bug missile
8010 CX=(1280+CX-64)MOD1280:CY=CY+32:IF CY>956 THEN BC=0
8020 IF POINT(CX+32,CY-16)=0 THEN RETURN
```

```
8100 REM Missile Hit
8110 IF POINT(CX+32,CY-16)<>1 THEN RETURN
8120 GOTO 6510
8500 REM Print Score
8510 PRINTCHR$30;"SCORE :";SCORE:GCOL0,0:RETURN
9000 REM Sound Effects
9020 SOUND0000,1,100,200:RETURN
9799 REM Envelopes
9800 ENVELOPE1,30,0,0,0,0,0,0,45,-10,-10,-10,126,126
9810 ENVELOPE3,1,0,0,0,0,0,0,45,-10,-10,-10,126,126:RETURN
9999 END
```

CONTINUATION FROM P.19

```
>
>L.
   10  ENVELOPE 1,5,1,-1,0,30,30,0,80,-5,-1,-2,120,80
   20  SOUND 1,1,60,20
   30  SOUND 2,1,90,20
   40  SOUND 3,1,115,20
   50 INPUTA$:GOTO20
>
>
>L.
    5 X%=10
   10 ENVELOPE1,5,10,0,0,40,0,0,120,-60,-5,-3,120,70
   15 REPEAT X%=X%+10
   20   SOUND1,1,X%,15
   30   UNTILO
>L.
   10 ENVELOPE 1,2,8,-8,8,1,1,1,120,0,0,0,120,0
   15 REPEAT
   20   SOUND 1,1,50,5
   30   UNTILO
>
```

98-100 The Albany,
Old Hall Street,
Liverpool L3 9EP.
Tel: 051-227 2642

BUG-BYTE SOFTWARE

CHESS

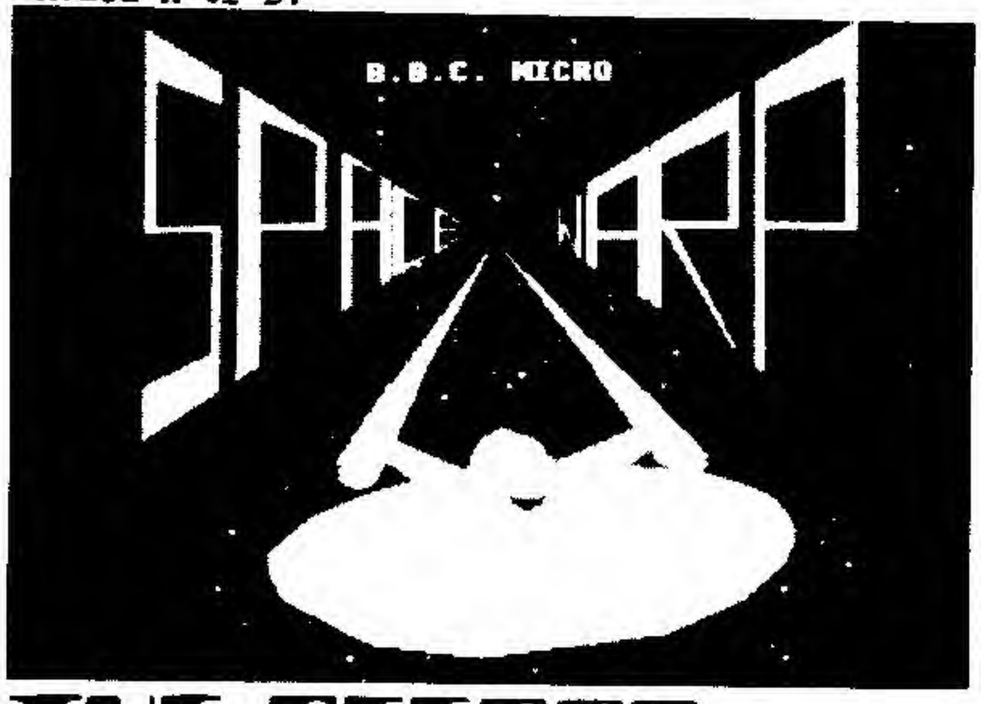
A fast version with many levels of play, using high resolution display. There are many options, including setting up, saving games on tape, changing level of play, castling and "en passen". Plays a very strong game. Price £11.50 inclusive.
This program is suitable for the Model A(32K RAM) and the Model B.

GOLF

A very true to life simulation of an eighteen hole course. Complete with fairways, bunkers, greens and streams. Altogether a very compulsive game.
Price £7.00 inclusive. Suitable for the Model A(32K RAM) and the Model B.

MULTIFILE

A multi-purpose filing system, with user definable set up. It caters for all file handling, including modifying, fast search, extensive calculation facilities and more. Price £25.00 inclusive.
Model A or B.



SPACE WARP

The ultimate Star Trek type game, extremely complex, with high resolution colour graphics and sound. It is supp-lied complete with a sixteen page manual and function key labels. It requires 32K RAM. Price £11.50 inclusive.
Suitable for the Model A(32K RAM) and the Model B.

ACTUAL SCREEN PHOTOGRAPH

THE BEEBON

A new magazine dedicated exclusively to users of the B.B.C. Micro. The Beebon contains tested programs, features on programming, special offers, features on hardware, reviews and much more. There will be at least three substantial programs in each issue, written by professional programmers. Published every two months, starting in May.
Annual subscription (6 issues) only £7.50.

BACKGAMMON

A standard Backgammon game, for the Model A(32K RAM) and the Model B B.B.C. Micros . It has extremely fast computer resposes. Full playing instructions are included. Price £8.00.

PLEASE SUPPLY. . . . . .

NAME: . . . . . .

ADDRESS: . . . . . .

I ENCLOSE A CHEQUE/POSTAL ORDER . . . . . .

OR PLEASE DEBIT MY ACCESS/BARCLAYCARD NUMBER. . . . . .

SIGNATURE. . . . . .

SEND TO: BUG-BYTE, 100 THE ALBANY, OLD HALL STREET, LIVERPOOL L3 9EP.

BUG-BYTE
SOFTWARE